

object.py

```
1  from utils import *
2  from shape3d import Vertex, Shape2D, Shape3D
3
4  class Object:
5      def __init__(self,
6          _position : Vertex = Vertex(0, 0, 0),
7          _face : Vertex = Vertex(0, 0, 1),
8          _shape : Shape2D | Shape3D = None,
9          _edge : float = 1
10         ) -> None:
11         self.position : Vertex = _position
12         self.face : Vertex = _face
13         self.translation : list[float] = None
14         self.rotation : tuple[float, str] = None
15         self.shape : Shape2D | Shape3D = _shape
16         self.edge : float = _edge
17
18     def __repr__(self) -> str:
19         return f"POS: {self.position}\npos: {self.shape.position}\nFACE: {self.face}\nT:
{as_colored(self.translation, fg=FOREGROUND_COLORS_STRONG['yellow'])}\nROT:
{as_colored(self.rotation, fg=FOREGROUND_COLORS_STRONG['red'])}\n{self.shape.position -
self.shape.vertices[0]}"
20
21     def push(self) -> None:
22         """
23         Give a random uniform vector of traslation
24         """
25         self.translation = [
26             uniform(*UNIFORM_DIRECTION_AREA),
27             uniform(*UNIFORM_DIRECTION_AREA),
28             uniform(*UNIFORM_DIRECTION_AREA)
29         ]
30
31     def add_translation(self, _vector : list[float]) -> None:
32         self.translation = _vector
33
34     def add_rotation(self, _angle : float, _axis : str) -> None:
35         self.rotation = (_angle, _axis)
36
37     def draw_vector(self, _vector):
38         glLineWidth(2)
39         glBegin(GL_LINES)
40         glColor3f(*COLORS['cyan'])
41         glVertex3f(self.position.x, self.position.y, self.position.z)
42         glVertex3f(self.position.x + _vector.x, self.position.y + _vector.y,
self.position.z + _vector.z)
43         glEnd()
44         glLineWidth(1)
45
46     def draw_y_axis(self):
47         glLineWidth(2)
48         glBegin(GL_LINES)
49         glColor3f(*COLORS['red'])
50         glVertex3f(self.position.x, self.position.y+1, self.position.z)
51         glVertex3f(self.position.x, self.position.y-1, self.position.z)
52         glEnd()
53         glLineWidth(1)
```

```

54
55     def draw_center(self):
56         glPointSize(8)
57         glBegin(GL_POINTS)
58         glColor3f(*COLORS['white'])
59         glVertex3f(self.position.x, self.position.y, self.position.z)
60         glEnd()
61         glPointSize(1)
62
63         glPointSize(8)
64         glBegin(GL_POINTS)
65         glColor3f(*COLORS['cyan'])
66         glVertex3f(self.position.x + self.shape.position.x, self.position.y +
self.shape.position.y, self.position.z + self.shape.position.z)
67         glEnd()
68         glPointSize(1)
69
70     def draw(self, _draw_edges : bool = False, _draw_vertices : bool = False) -> None:
71
72         self.shape.draw(self.position, _draw_edges, _draw_vertices)
73
74     def translate(self) -> None:
75         if self.translation:
76             self.position.translate(self.translation)
77
78
79     def rotate(self) -> None:
80         """
81         Rotation in relation to Object Origin (Rotating only the object position and face
Vector).
82         """
83         if self.rotation:
84             if len(self.rotation) == 2:
85                 self.face.rotate(Vertex(0, 0, 0), *self.rotation)
86                 # self.position.rotate(self.position, *self.rotation)
87                 self.shape.rotate(Vertex(0, 0, 0), *self.rotation)
88             elif len(self.rotation) == 3:
89                 self.face.rotate_any(Vertex(0, 0, 0), *self.rotation)
90                 # self.position.rotate_any(self.position, *self.rotation)
91                 self.shape.rotate_any(Vertex(0, 0, 0), *self.rotation)
92
93
94     def spin(self):
95         self.rotation = [
96             uniform(*UNIFORM_DIRECTION_AREA),
97             uniform(*UNIFORM_DIRECTION_AREA),
98             uniform(*UNIFORM_DIRECTION_AREA)
99         ]
100
101     def collide(self):
102         if self.translation:
103             collision = [1, 1, 1]
104             if self.position.x > BORDER_COLLITION_CAP or self.position.x < -
BORDER_COLLITION_CAP:
105                 collision[0] = -1
106             if self.position.y > BORDER_COLLITION_CAP or self.position.y < -
BORDER_COLLITION_CAP:
107                 collision[1] = -1
108             if self.position.z > BORDER_COLLITION_CAP or self.position.z < -
BORDER_COLLITION_CAP:
109                 collision[2] = -1

```

```
110
111         self.translation = [
112             self.translation[0] * collision[0],
113             self.translation[1] * collision[1],
114             self.translation[2] * collision[2]]
115
116
117 if __name__ == '__main__':
118     pass
119
```